Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-milestone-4-trivia-2024-m24/grade/vvh

Course: IT114-003-F2024
Assigment: [IT114] Milestone 4 Trivia 2024 M24
Student: Valeria C. (vvh)

Submissions:

Submission Selection

1 Submission [submitted] 12/9/2024 11:08:16 PM

Instructions

COLLAPSE

- Implement the Milestone 4 features from the project's proposal document: https://docs.google.com/document/d/1h2aEWUoZ-etpz1CRl-StaWbZTjkd9BDMq0b6TXK4utl/view
- Make sure you add your ucid/date as code comments where code changes are done
- All code changes should reach the Milestone4 branch
- Create a pull request from Milestone4 to main and keep it open until you get the output PDF from this assignment.
- Gather the evidence of feature completion based on the below tasks.
- Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
- Run the necessary git add, commit, and push steps to move it to GitHub
- Complete the pull request that was opened earlier
- Upload the same output PDF to Canvas

Branch name: Milestone4

Group

100%

Group: Away
Tasks: 1
Points: 2.5

COLLAPSE

Task

100%

Group: Away
Task #1: Away: Client can mark themselves “away” to be skipped in the turn flow but still be in the game
Weight: ~100%
Points: ~2.50

^ COLLAPSE ^

Details:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

Columns: 4

Sub-Task

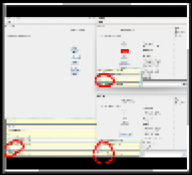
100%

Group: Away
Task #1: Away: Client can mark themselves

Task Screenshots

Gallery Style: 2 Columns

4 2 1



player marked himself away and shows to other clients but is still in the game

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Sub-Task

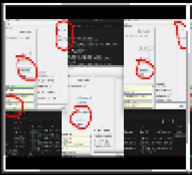
100%

Group: Away
Task #1: Away: Client can mark themselves

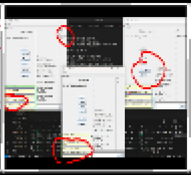
Task Screenshots

Gallery Style: 2 Columns

4 2 1



status updating to all clients and how it displays to the client



shows when user is away/not away

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Away
Task #1: Away: Client can mark themselves

Task Screenshots

Gallery Style: 2 Columns

4 2 1

two players set themselves away but they are still in the game

player just marked herself not away but is still in the game and will keep playing on next round

player marks herself not away and its back to the

Sub-Task

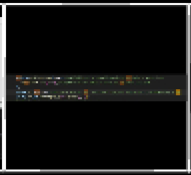
100%

Group: Away
Task #1: Away: Client can mark themselves

Task Screenshots

Gallery Style: 2 Columns

4 2 1



This happens when the client user interacts with the UI element in the GamePanel

The GameRoom's handleAway method updates the player's state and notifies all clients

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

game on
round 5

Caption(s) (required) ✓
Caption Hint:
Describe/highlight what's being shown

Explain in concise steps how this logically works
Response:

The "away toggle" begins with a client interacting with the UI by clicking the button, which triggers the client-side logic to construct and send an "AWAY" or "NOT_AWAY" payload to the server. The server receives this payload, determines the intended status (away or not away), and forwards it to the current game room. The game room updates the player's state and broadcasts the updated "away" status to all connected clients.

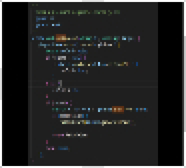
Sub-Task 100% Group: Away Task #1: Away: Client can mark themselves

Task Screenshots

Gallery Style: 2 Columns

4 2 1

message is created and broadcast to all clients using the sendGameEven method in the server's GameRoom

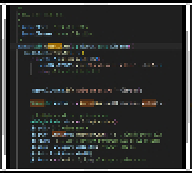

Iterates over the connected clients, sending the event string via the sendGameEven... method in the ServerPlayer

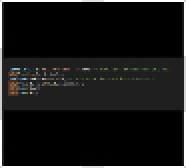
Sub-Task 100% Group: Away Task #1: Away: Client can mark themselves

Task Screenshots

Gallery Style: 2 Columns

4 2 1


userlistpanel adding user to the list panel


adds user to the user list

client ui

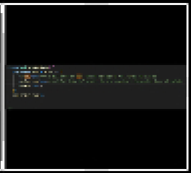
Sub-Task 100% Group: Away Task #1: Away: Client can mark themselves

Task Screenshots

Gallery Style: 2 Columns

4 2 1

The handleAway method in the GameRoom class updates the player's "away" status and notifies all players about


resets away status

class or ServerThread

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The process begins on the server side, where a game event, such as a player locking in an answer, triggers a state update. The server creates a message detailing the event and broadcasts it to all connected clients using the sendGameEvent method. This method iterates through the clients, sending a payload with the GAME_EVENT type and the event message. On the client side, the payload is received, and the event message is parsed and displayed in the Game Events Panel through a method like addGameEvent

handles updating the user list by adding or removing user list

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The process of updating the User List Panel starts with the server sending a synchronization payload to the client when user-related events either by joining or leaving a room occur. On the client side, the onSyncClient method processes this payload by invoking the addUserListItem method in the UserListPanel. This method checks if the user already exists in the list; if not, it creates a new UserListItem with the provided clientId and clientName

the change

away status update

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The logic for skipping away players works by first updating and notifying the server and all players when a player toggles their away status. During gameplay, the server checks a player's status before allowing them to take a turn or submit an answer. If a player is marked as away, their actions, such as attempting a turn or submitting an answer, are ignored and they are notified that they cannot participate while away

End of Task 1

End of Group: Away
Task Status: 1/1

Group

100%

Group: Spectator
Tasks: 1
Points: 2.5

COLLAPSE

Task

100%

Group: Spectator
Task #1: Spectator: Client can join as spectator
Weight: ~100%
Points: ~2.50

COLLAPSE

Details:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.
Spectator control/access logic must be handled in the GameRoom.
They can see all chat but are ignored from turns and can't send messages

Columns: 4

Sub-Task

100%

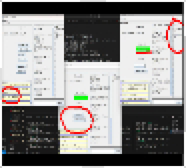
Group: Spectator Task #1: Spectator: Client can join as spectator

Task Screenshots

Gallery Style: 2 Columns

4 2 1

room java handling spectate function ensuring it is only processed when spectate action is activated



showing how spectators cannot select answer

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's

Sub-Task

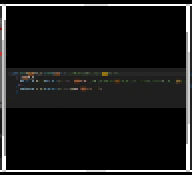
100%

Group: Spectator Task #1: Spectator: Client can join as spectator

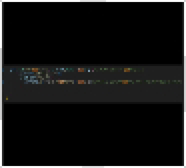
Task Screenshots

Gallery Style: 2 Columns

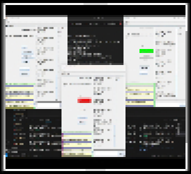
4 2 1



User List Panel represent spectators from active players.



it updates the user spectator status from active players



panel representation as a spectator

Sub-Task

100%

Group: Spectator Task #1: Spectator: Client can join as spectator

Task Screenshots

Gallery Style: 2 Columns

4 2 1

gameroom spectating status

room java handling spectate function ensuring it is only processed when spectate action is activated

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's

Sub-Task

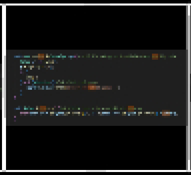
100%

Group: Spectator Task #1: Spectator: Client can join as spectator

Task Screenshots

Gallery Style: 2 Columns

4 2 1



sending payload to update spectating status

payload type handling spectate or not spectating

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

Task Response Prompt

being shown

Caption(s) (required) ✓

Caption Hint:

Describe/highlight what's being shown

being shown

Task Response Prompt

Explain in concise steps how this logically works

Response:

The code logic to block or ignore spectators from game actions, such as taking turns or sending specific game-related messages, works by checking each player's spectator status before processing their actions. When a player initiates an action, like taking a turn, the server-side logic, handleTurn, first verifies if the player is actively participating by checking their isSpectating flag. If the flag is true, the server bypasses or rejects the action, often sending a message back to the player indicating they cannot participate while spectating

Prompt

Explain in concise steps how this logically works

Response:

When a player toggles their spectator status, the server processes this change via the spectate method, which updates the player's spectator state and broadcasts it to all connected clients using the sendSpectateStatus method. Each client receives a payload containing the updated spectator status and processes it in the processSpectatePayload method, which delegates the update to the UserListPanel. The UserListPanel then updates its UI by modifying the relevant UserListItem, appending or removing the "SPECTATOR label based on the player's current status.

End of Task 1

End of Group: Spectator
Task Status: 1/1

Group

100%

Group: Project Specific
Tasks: 3
Points: 4

COLLAPSE

Task

Group: Project Specific
Task #1: Spectator Extra

100%

Task #1: Spectator Extra
Weight: ~33%
Points: ~1.33

COLLAPSE

Details:
Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

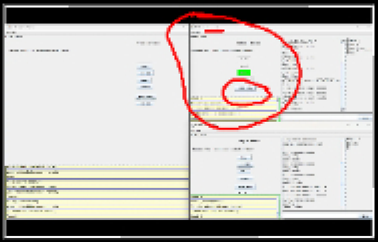
Columns: 2

Sub-Task
100%
Group: Project Specific
Task #1: Spectator Extra
Sub Task #1: Show that a spectator can see the correct answer

Task Screenshots

Gallery Style: 2 Columns

4 2 1



showing how player 3 (Nicole) can see the answer of the questions as an spectator

Caption(s) (required) ✓

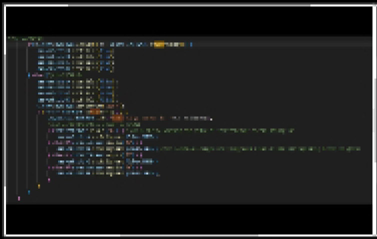
Caption Hint: *Describe/highlight what's being shown*

Sub-Task
100%
Group: Project Specific
Task #1: Spectator Extra
Sub Task #2: Show the code related to sending/showing the correct answer to spectators

Task Screenshots

Gallery Style: 2 Columns

4 2 1



spectating showing the correct answers to spectators

spectate status button

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

When the user toggles the spectate button, the application determines whether the user is currently spectating by checking the Client.INSTANCE.isSpectating() status. If spectating, clicking the button sends a NOT_SPECTATE payload to the server, updates the button text to "Spectate," and re-enables the answer buttons to allow participation. If not spectating, clicking the button sends a SPECTATE payload to the server, disables all answer buttons to prevent participation in gameplay, and sends an "AWAY" status to the server to mark the user as non-participating

End of Task 1

Task

100%

Group: Project Specific
Task #2: Add option to choose which categories will be used for the session
Weight: ~33%
Points: ~1.33

^ COLLAPSE ^

Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

Columns: 2

Sub-Task

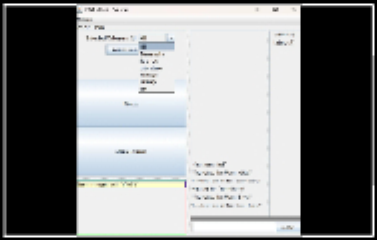
100%

Group: Project Specific
Task #2: Add option to choose which categories will be used for the session
Sub Task #1: Show the pre-game screen (can be ready panel) that provides the option for category selection

Task Screenshots

Gallery Style: 2 Columns

4 2 1



UI screen showing in pregame phase the category selection option

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

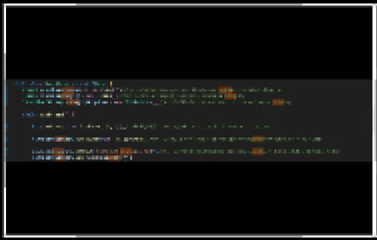
100%

Group: Project Specific
Task #2: Add option to choose which categories will be used for the session
Sub Task #2: Show the code that sets and uses the chosen categories in the GameRoom

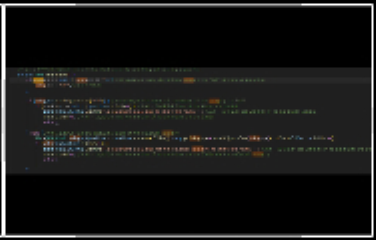
Task Screenshots

Gallery Style: 2 Columns

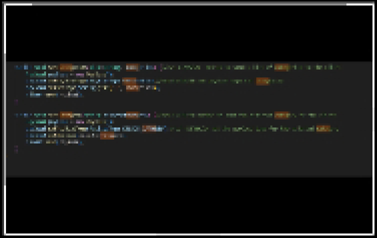
4 2 1



select category button



gameroom code showing categories selection



serverplayer java code handling categories selection

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The logic for handling chosen categories in the GameRoom starts with the setCategory method, where the selected category is updated, a notification is sent to all players, and the category is communicated to the clients. During gameplay, the startRound method checks if a category is set; if none, it defaults to "All". Questions are filtered based on the chosen category or left unfiltered if "All" is selected. A random question is selected from the filtered list, sent to all players, and removed from the master list to avoid repetition. The getQuestionCategories method ensures all unique categories are extracted from the available questions and sent to clients using sendCategories.

End of Task 2

Task

100%

Group: Project Specific
Task #3: Add ability to add new questions
Weight: ~33%
Points: ~1.33

COLLAPSE

Details:

Screenshots of editors must have the frame title visible with your ucid and the client name.
Code screenshots must have ucid/data comments.

Columns: 3

Sub-Task

100%

Group: Project Specific
Task #3: Add ability to add new questions
Sub Task #1: Show the pre-game

Sub-Task

100%

Group: Project Specific
Task #3: Add ability to add new questions
Sub Task #2: Show the code that

Sub-Task

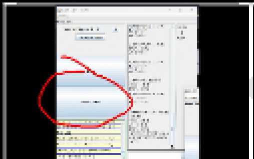
100%

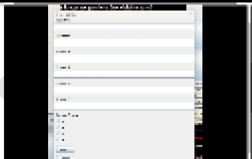
Group: Project Specific
Task #3: Add ability to add new questions
Sub Task #3: Show a before and after

Task Screenshots

Gallery Style: 2 Columns

4 2 1

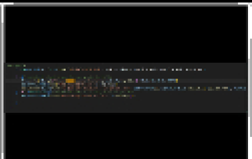


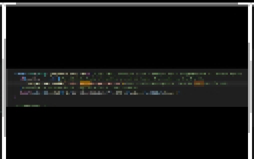


Task Screenshots

Gallery Style: 2 Columns

4 2 1

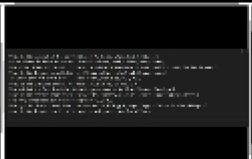


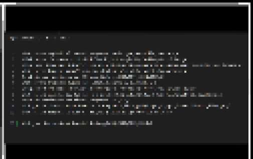


Task Screenshots

Gallery Style: 2 Columns

4 2 1





pre-game screen showing add question button	screen that pops up when add question is pressed	related code handling new questions	gets the questions and loads them under their respective category	questions.txt before adding question	questions.txt after question added

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

When a new question is submitted, the addQuestion method is invoked, receiving a Payload object containing the question details. This payload is cast to an AddQuestionPayload object, which includes the question text, category, answer options, and the correct answer. The saveQuestionToFile method formats these details into a comma-separated string and appends it to the questions.txt file using Files.write with StandardOpenOption.APPEND, ensuring the question is saved persistently without overwriting existing entries. Once saved, the server broadcasts a message to all players in the room using sendGameEvent, notifying them of the new addition along with the player's ID and name. Any errors during file operations are logged using LoggerUtil

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The new question is saved to the questions.txt file in a structured manner using the saveQuestionToFile method. First, the server extracts the question's details, including the text, category, four answer options, and the correct answer, from the AddQuestionPayload object. These details are formatted into a single line as a comma-separated string, ensuring all necessary information is captured in a consistent format. For example: QuestionText,Category,AnswerA,AnswerB

This string is appended to the questions.txt file using the Files.write method with the StandardOpenOption.APPEND option, ensuring the new data is added to the end of the file without overwriting its existing content. If the operation is successful, a log entry is made confirming the addition; if an error occurs, it is logged for debugging

End of Task 3

End of Group: Project Specific
Task Status: 3/3

Group

100%

Group: Misc
Tasks: 3

100%

Points: 1

^ COLLAPSE ^

Task

100%

Group: Misc
Task #1: Add the pull request link for the branch
Weight: ~33%
Points: ~0.33

^ COLLAPSE ^

ⓘ Details:

Note: the link should end with /pull/#

Task URLs

URL #1

https://github.com/vvh24/vvh-IT114-003/pull/14

URL

https://github.com/vvh24/vvh-IT114-003/pull/14

End of Task 1

Task

100%

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

^ COLLAPSE ^

Task Response Prompt

Response:

For this milestone, I could solve the problem of why the questions werent displaying at it was because I was not putting the right path to my questions file. After I made the adjustments the code did work and I worked on commenting what what added and implemented to it. Just as milestone 3 since I wanted to integrate milestone 3 and 4 so I didnt have to be back and forth, I had a lot of problems at making the away and spectate functions to work because I did not know how to make it work, but I had to do some research and ask to other partners to understand the logic behind it and try to implement it to my program. The only issue I have is adding the question. for some reason, I am able to add a new question and it goes to the file, but it doesnt registered in the game once its created. I am not sure if this is because it skips a line when the question is created because when I get rid of the line, the questions load normally. I am not sure if its because the client id 0 suggests that the client connection isnt associated with a valid player which may be due to an issue during the payload construction where the client ID is missing something? and or there can be an issue in how the add_question payload is constructed on the client side

End of Task 2

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

COLLAPSE

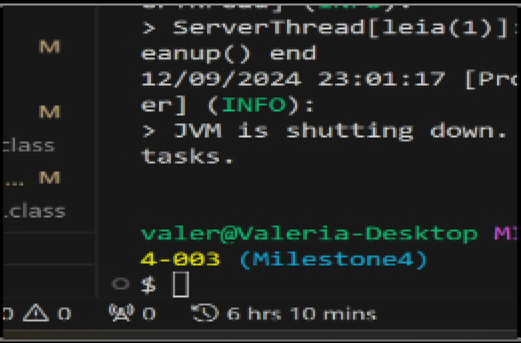
Details:

Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots

Gallery Style: 2 Columns

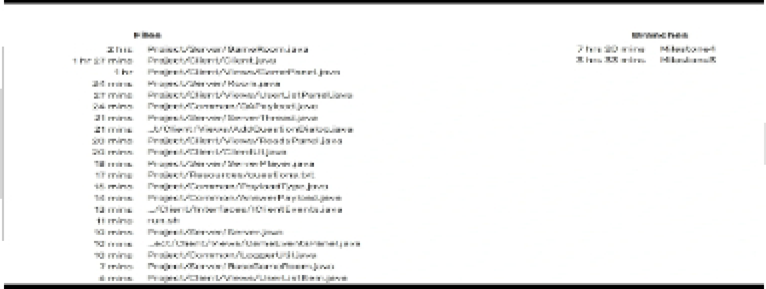
4 2 1



```
> ServerThread[leia(1)]:
eanup() end
12/09/2024 23:01:17 [Pro
er] (INFO):
> JVM is shutting down.
tasks.

valer@Valeria-Desktop MI
4-003 (Milestone4)
$ 
```

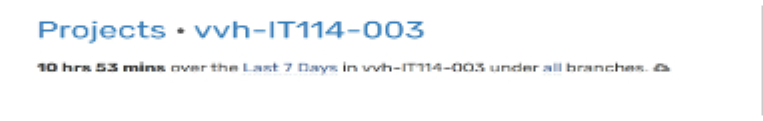
vscode registering time from wakatime



Files	
2 hrs	Project/Server/GameRoom.java
1 hr 27 mins	Project/Client/Client.java
1 hr	Project/Client/Views/GamePanel.java
24 mins	Project/Server/Room.java
27 mins	Project/Client/Views/UserListPanel.java
24 mins	Project/Common/QAPayload.java
21 mins	Project/Server/ServerThread.java
21 mins	.../Client/Views/AddQuestionDialog.java
20 mins	Project/Client/Views/ReadyPanel.java
20 mins	Project/Client/ClientUI.java
18 mins	Project/Server/ServerPlayer.java
17 mins	Project/Resources/questions.txt
16 mins	Project/Common/PayloadType.java
14 mins	Project/Common/AnswerPayload.java
12 mins	.../Client/Interfaces/IClientEvents.java
11 mins	run.sh
10 mins	Project/Server/Server.java
10 mins	...ect/Client/Views/[illegible].java
10 mins	Project/Common/LoggerUtil.java
7 mins	Project/Server/BaseGameRoom.java
6 mins	Project/Client/Views/UserListItem.java

Branches	
7 hrs 20 mins	Milestone4
3 hrs 33 mins	Milestone3

showing time between files and branches



Projects • vvh-IT114-003

10 hrs 53 mins over the Last 7 Days in vvh-IT114-003 under all branches.

overall time over the last 7 days

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment